

Python Fun(damentals)

Python Data Structures

Alexander Rymdeko-Harvey

Obscurity Labs

```
* list  
* dict
```



What Are Data Structures

We will not be covering all types as the scope of this course is a intro to the CS / Python world.

- Non-primitive types are part of the data structure family.
 - arrays
 - lists
 - files
- Traditional computer science world requires a way to store data.
- Data structures help programmers store data for later use, pass data, store collections etc.
- Critical to the design of how programmers process data.
- Python provides many features that would be required to work with these structures.

Python3 Lists

Using the python interpreter we can perform live data struc operations:

```
Python 3.7.5 (default, Apr 19 2020, 20:18:17)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = [1, 2, 3, 4] # list of ints

>>> type(x) # check the python type
<class 'list'>
```

What does this look like under the hood:

Each entry is stored in a stack structure, imagine a large `todo` list at home:

1	2	3	4
---	---	---	---

With this you can reference on of those `todo` items using the simple index operator:

```
Python 3.7.5 (default, Apr 19 2020, 20:18:17)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
#      0, 1, 2, 3 <--- Index location
>>> x = [1, 2, 3, 4]
>>> x[2] # ref the second entry in the list
3
```

Python3 lists unlike entries

One of the unique factors of the `list` is there ability to store multiple unlike types. Unlike a traditional array where all values must be the same type.

```
Python 3.7.5 (default, Apr 19 2020, 20:18:17)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
#      0,    1,    2,    3 <--- Index location
>>> x = [1, 'hi', 'bye', 4]
>>> x[2] # ref the second entry in the list
hi
```

Python3 Lists Mutable Properties

Lists are mutable, which means that you can change their content without changing their identity.

```
Python 3.7.5 (default, Apr 19 2020, 20:18:17)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
#      0,      1,      2,      3 <--- Index location
>>> x = [1, 'hi', 'bye', 4]

>>> x[2] = 'fun' # change the entry to a new value

>>> print(x)
[1, 'hi', 'fun', 4]
```

Lab 1 - Familiarization

Tasking

Using the new `python` command perform the following get familiar with `list` types:

1. Create a basic list ex. `[1, 'hi', 'bye', 4]`
2. Attempt to change a value in this list by its index value

Python3 dict

Python Dictionaries are exactly what you need if you want to implement something similar to a telephone book. Many of the other structures don't provide you the capability needed to store a state.

```
Python 3.7.5 (default, Apr 19 2020, 20:18:17)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = {'alex': '215-586-1111', 'megan': '215-586-1111', 'daisy': '', 'joe': ''}

>>> type(x)
<class 'dict'>
```


Working with Python dict

Python Dictionaries are similar to lists as they are mutable and values can be changed. but unlike list you would need to know where every entry is within the list to look up a phone number right?

Python dict provides a lookup table to handle this:

```
Python 3.7.5 (default, Apr 19 2020, 20:18:17)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = {'alex': '215-586-1111', 'megan': '215-586-1111', 'daisy': '', 'joe': ''}

>>> x['alex']
'215-586-1111'
```

These are called keys , many technologies use a concept called keystore . Redis is a example of this and they are generally in-memory data stores for fast and random access to your data.

Python3 dict Mutable Properties

`dict` are mutable, which means that you can change their content without changing their identity.

```
Python 3.7.5 (default, Apr 19 2020, 20:18:17)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = {'alex': '215-586-1111', 'megan': '215-586-1111', 'daisy': '', 'joe': ''}

>>> x['alex'] = "215-000-1111"

>>> x['alex']
'215-000-1111'
```

Lab 2 - Familiarization

Tasking

Using the new `python` command perform the following get familiar with `dict` types:

1. Create a basic dict ex. `{'alex': '215-586-1111', 'megan': '215-586-1111', 'daisy': '', 'joe': ''}`
2. Attempt to change a value in this dict by its key value name.